U

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/901,254 | 07/09/2001 | Peter F. Sollich | MS174298.1 | 7787 |

27195 7590 08/03/2004

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

| EXAMINER | |
|---|---|
| VU, TUAN A | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2124 | |

DATE MAILED: 08/03/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) |
|---|---|---|
| | 09/901,254 | SOLLICH, PETER F. |
| | Examiner | Art Unit |
| | Tuan A Vu | 2124 |

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *09 July 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *09 July 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *2002/02/12*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

## DETAILED ACTION

1. This action is responsive to the application filed July 9, 2001.

Claims 1-40 have been submitted for examination.

### *Oath/Declaration*

2. The oath or declaration is defective, i.e. without signature execution. A new oath or declaration in compliance with 37 CFR 1.67(a) identifying this application by application number and filing date is required. See MPEP §§ 602.01 and 602.02. It was not executed in accordance with either 37 CFR 1.66 or 1.68.

It has been indicated that a Notice of missing parts ( for declaration without proper signature) was mailed 10/26/01. But the expected corrected and resubmitted Declaration and Power of Attorney is found to be still missing in the present application file. Examiner would suggest that Applicant re-submit the above presumed corrected Declaration and Power of Attorney when Applicants file the response to this current Office Action.

### *Claims objections*

3. Claim 17 is objected to because of the following informalities: there appears to be a grammatical error in the use of the element "... and access<u>ed</u>..." in line 2. Examiner will see this as being "and access..." to examine the merits of the claims.

4. Claim 3 is objected to because the element 'comb-vector' (line 1) is not recited in claim 1; hence there appears be a typo error in reciting dependency on 'system of claim 1' ( line 1); and this should be 'system of claim 2' lest a 35 USC 112, 2nd paragraph type of rejection would apply. Appropriate correction is required.

### *Claim Rejections - 35 USC § 103*

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-12, 14, 16-20, 22-35, 37-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bottomley, USPubN: 2004/0015912 ( hereinafter Bottomley), in view of Driesen et al., "Selector Table Indexing & Sparse Arrays", October 1993(hereinafter Driesen).

**As per claim 1**, Bottomley discloses a system for facilitating an interface dispatch, comprising:

a pre-execution engine adapted to load source code ( pg. 5, para 0032-0039 – Note: class loading and resolving of method invocation is equivalent to pre-execution engine) and allocate a block of memory in the form of vector for creating an interface map (e.g. *master interface table* – para 0039 – pg. 5 ) that includes a plurality of slots ( e.g. *MITABLE 152* - Fig. 14) for referencing interface virtual tables (e.g. IVTABE 180 – Fig. 14 ) that correspond to an implementation of an interface in a class type (e.g. *interfaces the class implements* – para 39, pg. 5; *for each class, the MItable 152 ...* – para 0069, pg. 8); and

an interface index component (e.g. system interface table 150 – Fig. 14) adapted to assign index numbers to interfaces ( e.g. *1 to J* – para 0069, pg. 8) are loaded by the pre-execution engine (PEE); the PEE adapted to determine row structure ( Note: an entry in the table 152 Fig. 14 is a row structure) for a class type and interfaces implemented by the class type using the indices assigned to the interfaces and associating indices in the interface map (IM) based on the row structure and storing references (e.g. *Interface(J-1) <-> Ref. to Ivtable( J-1)* – Fig. 14 ) to the interface virtual tables (IVT).

But Bottomley does not explicitly disclose associating indices with empty slots in the IM based on the row structure configuration and storing references to the IVT in the empty slots when enough empty slots are found for respective row structure. The allocating of table, vector or array size for dispatching interface method invocation with a redundant allocation of memory therefor in order to provide for unpredictable number of methods invocation for implementing an interface in OO program execution was a known concept at the time the invention was made, i.e. more than one similar class method signature implementing similar interfaces and no all method signatures being implemented for a given interface. To that effect, Bottomley discloses empty slots left blank ( NULL – Fig. 14; *elided* - para 0069 – pg. 8) in the MITable. Further, Driesen, in a method for using dispatch table for looking up methods implementing a class being loaded in a Java runtime environment similar to the class loading and method resolution by Bottomley, discloses empty slots in a single row structure or array that make up the 2-dimensional table Class/method selector (Fig. 4, chp. 3.2 ); hence has implicitly disclosed leaving empty slots in associating method indices with a row structure and filling these slots when slots are still available and as long as more methods are called for implementing a particular class. In case Bottomley does not associate indices with empty slots in the IM when slots are found for a particular class type row, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the empty slots and the association of interface indices with the allotted slots as taught by Driesen because of the reason known in the art of providing redundant row storage ( as suggested by Bottomley and taught by Driesen) in the interface map structure in view of the unpredictable number of methods needed to be invoked when implementing an interface as mentioned above in a object-oriented program pre-execution method resolution process, and providing this redundancy of empty slots would alleviate memory conflicts problem.

**As per claim 2**, Bottomley does not disclose creating a row-structure with a comb-vector technique; but Driesen disclose a combination of row structure or vectors with empty slots to form a final IM structure; thus, teaches a equivalent of a comb-vector technique. The rationale as to modify Bottomley empty slots table so to apply the comb-vector as suggested by Driesen would have been obvious because this allow the creation of row structure in a as-needed basis and thus alleviate memory resources usage, i.e. allow array to shift or grow as needed (see Driesen – chp. 3.4.2, 3.4.3; Fig. 8-9)

**As per claim 3**, Bottomley only teaches eliding of row until hitting a row allotted for a method (*elided* - para 0069 – pg. 8 ); and Driesen teaches row structure that can expand linearly and insertion is possible whenever room is needed into such structure ( ch. 3.2 ); hence, both teach sliding row structures until each interface entry hits an empty slot.

**As per claim 4**, Bottomley does not explicitly teach assigning a row start location and providing offsets from such start location. But Bottomley assign a first non-Null entry in a sutructure build upon the concept of row incrementing ( *MItable 152* – Fig. 14). Official notice is taken that an array or vector being allotted with a starting index is equivalent to every elements of the array being but some sequential offsets from that starting position. Hence, in view of such notice, Bottomley has implicitly disclosed a starting row location with other index interface numbers or indices being offsets from that first location.

**As per claim 5**, Bottomley does not explicitly disclose that a row start location for a class type is same for another class type. But in view of the teachings that each MITABLE is created for a particular class (e.g. *interfaces the class implements* – para 39, pg. 5;*for each class, the MItable 152 ...* – para 0069, pg. 8), this limitation is strongly suggested. In view of the teachings by Driesen to use an array per class type (Fig. 3- ch. 3.2), it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a row structure per class so that a start location for that class type would be same, as taught by Driesen, to the creating of row in Bottomley, because this would distinguish class type from one another and possibly alleviate memory fragmentation and unless storage conflict dictates otherwise, this type-related spatial grouping would expedite process for dispatching a method coming from a same class implementing an interface, such quickening process being the primary intention of Bottomley's dispatch and resolution system.

**As per claim 6**, Bottomley discloses a loader and a linker (class loader 134 – Fig. 5; Fig 11-15).

**As per claims 7 and 8**, Bottomley discloses a JVM with interpretation of byte code and binding methods using constant pool ( Fig. 3, 9). Official notice is taken that the use of a Java machine with optimization and interpretation capability just before execution time such as a JIT was a known concept at the time the invention was made; and JIT would be known to interpret, recompile and execute. So, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a compiler like the JIT to Bottomley's method for using a JVM with quickening techniques for loading and resolving methods invocation because of the benefits ( as known in the art ) provided by the JIT in that it can recompile after code optimization, and interpreting when compilation is no required, thus alleviate resources for runtime.

**As per claim 9**, Bottomley assign indices into the table sequentially (Note: this is inherent in programming language for allocating data to a vector or array )

**As per claim 10**, this claim correspond to the limitation of claim 4, hence is rejected using the rejection as set forth therein.

**As per claim 11**, Bottomley does not explicitly disclose a reference to an interface being computed by adding the index number to the desired interface to the row start location. But, in view of the rationale from claim 4, a reference to an interface would be a start index, e.g. 0, being added with the increment offset ( *J-1* – Fig. 14) created from that start location index; hence this limitation is implicitly disclosed.

**As per claim 12**, Bottomley disclose a row being stored in a method table corresponding to class type (e.g. *table 180* – Fig. 14)

**As per claim 14**, Bottomley does not explicitly disclose creating additional IMs when the current IMs are filled. But the intention for Bottomley to create MITable for the context of dynamically resolving methods ( re claim 1) implementing interfaces suggests that as needed, more MItable are created. Hence, the limitation as to create additional IMs would have been obvious in view of the dynamic need to resolve the methods invoked at binding time, whenever the resources are available.

**As per claim 16**, Bottomley discloses receiving a method call of an interface and access the interface virtual table using the interface map (e.g. IVTable, MItable - Fig. 14).

**As per claim 17**, this claim includes referencing an interface using a row start location and an index corresponding to an interface, which has been similarly recited and addressed in claims 4 and 10.

**As per claim 18**, Bottomley discloses a method of creating a reference map to interface virtual tables ( IVTs), comprising:

allocating a vector in memory (Note: an entry in the table 152 Fig. 14 is a row structure) of a predetermined size ( Note: a array is equivalent to being declared with a predetermined size) having a plurality of slots for creating an interface map for storing references to IVTs (e.g. *MITable 152* - Fig. 14);

loading source code having plurality of classes and interfaces; assigning indices to the interfaces (e.g. Fig. 13; System Interface 150; pg. 5, para 0032-0039 – Note: each entry in the System Interface table represent an index);

determining a row structure for a class type and interfaces implemented by the class type utilizing the indices (e.g. *interfaces the class implements* – para 39, pg. 5;*for each class, the MItable 152 ...* – para 0069, pg. 8; *MITABLE 152* - Fig. 14); and

storing references to IVTs corresponding to the row structure in the plurality of slots (e.g. Fig. 14; para 0069, pg. 8).

But Bottomley does not disclose using a comb-vector technique; but this limitation has been addressed in claim 2 above.

**As per claim 19**, Bottomley discloses repeating the step of determining a row structure for a class type and interfaces implemented by the class type that implements interfaces in the source code (e.g. Fig. 14; para 0069, pg. 8)

**As per claim 20**, Bottomley does not explicitly disclose creating a second vector when the current IMs predetermined size is full. But the intention for Bottomley to create MITable for the context of dynamically resolving methods ( re claim 1) implementing interfaces suggests that as needed, more MItable are created. Hence, the limitation as to create additional vector in memory would have been obvious in view of the dynamic need to resolve the methods invoked at binding time, whenever the resources are available.

**As per claims 22-23**, refer to rejection of claims 3-4, respectively.

**As per claim 24**, this claim corresponds to claim 12, and is rejected with the rejection as set forth therein.

**As per claims 25-26**, these claims correspond to claims 5-10, respectively, and are rejected using the corresponding rejection set forth therein, respectively.

**As per claim 27**, Bottomley discloses utilizing a interface map ( MItable 152 – Fig . 14) to cast a class instance to an interface (Note: an interface cannot get instantiated unless through a class object, hence that is equivalent to casting such interface is via a class instance pointed by a IM).

**As per claim 28**, Bottomley discloses a medium having executable component comprising:

a pre-execution engine (PEE) component adapted to load source code having plurality of classes and interfaces;

assigning indices to the interfaces (e.g. Fig. 13; System Interface 150; pg. 5, para 0032-0039); assigning indices to the interfaces as they are loaded (e.g. Fig. 13; System Interface 150 Note: each entry in the System Interface table represent an index; pg. 5, para 0032-0039);

an interface map component including a plurality of slots for referencing IVTs that correspond to an implementation of an interface in a class type (e.g. *Ref. to Ivtable( J-1)* → *IVTABLE 4* – Fig. 14);

the PEE adapted to determine row structure ( Note: an entry in the table 152 Fig. 14 is a row structure) for a class type and interfaces implemented by the class type using the indices assigned to the interfaces and associating indices in the interface map (IM) based on the row structure and storing references (e.g. *Interface(J-1) <-> Ref. to Ivtable( J-1)* – Fig. 14 ) to the interface virtual tables (IVT).

But Bottomley does not explicitly disclose associating indices with empty slots in the IM based on the row structure configuration and storing references to the IVT in the empty slots when enough empty slots are found for respective row structure. But this limitation has been addressed in claim 1 above.

**As per claim 29**, Bottomley discloses execution engine to receive and execute code and access the IVT corresponding to the interface and class type utilizing the IM in response to a method call (e.g. Fig. 10-13) for a specific interface type implemented in a class instance of a specific class type (e.g. *Interface(J-1) <-> Ref. to Ivtable( J-1)* – Fig. 14).

**As per claim 30**, this claim is the execution engine version of claim 11; and is rejected using the same rejection set forth therein.

**As per claim 31**, refer to claim 3.

**As per claim 32**, this claim includes limitations of claims 4 and12; hence is rejected with the corresponding rejection as set forth therein respectively.

**As per claims 33 and 35**, refer to rejection of claims 9, and 14, respectively.

**As per claim 34 and 37**, Bottomley discloses creating a IM (e.g. *master interface table* – para 0039 – pg. 5), such map being a vector (Note: a Java type of Array for table rows is called a Vector).

**As per claim 38**, this is a system claim corresponding to method claim 18; and includes the means for performing the same steps limitations listed therein; hence is rejected using the corresponding rejection as set forth therein.

**As per claim 39**, this claim corresponds to claim 22; hence is rejected using the same rejection as set forth therein.

**As per claim 40**, Bottomley discloses means for accessing the references in the slots, for accessing the references using a row start location corresponding to a class type and adding the index number associated with the desired interface to the row start to access the reference to the IVT ( refer to corresponding rejection of claim 10 and 11)

7. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Bottomley, USPubN: 2004/0015912, and Driesen et al., "Selector Table Indexing & Sparse Arrays", as applied to claim 1, and further in view of Alpern, USPN: 6,652,248 (hereinafter Alpern).

**As per claim 13**, Bottomley does not teach that a slot is accessed by multiplying an index number by four and adding it to a row start location. But the storing of offsets list via memory implementation using a subset of a computer word (e.g. 1 byte out of 4 bytes) was a known concept in translation table for memory fast reference and retrieval of data stored outside of fast memory. For indexing for example, Bottomley suggests using index length of limited bits size (Fig. 8 ). Alpern, in a method using a reference table (IMT) to map method entries in a interface virtual table to a interface implemented by a class type, discloses a offsets value of limited size storage and adding of such offset to get to the interface method table ( e.g. *IMT OFFSET Field* - Fig. 2-5); hence has suggested a limited number of bytes within a word for allocating the range for offsets. If the architecture is a 32-bits memory bus, then it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide offset field being 4 bits as suggested by Alpern, and when adding it to the row start location as taught by Bottomley, multiply this offset by 4 before adding. This would cause the correct mapping of bit alignment with the scheme as laid out when the same memory is designed to produce a translation or mapping table as practiced in the art as exemplified by Alpern and the offset translation table.

8. Claims 15, 21, 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bottomley, USPubN: 2004/0015912, and Driesen et al., "Selector Table Indexing & Sparse Arrays", as applied to claim 1, 18, 34, and further in view of Miloushev et al., USPN: 6,226,6928 (hereinafter Miloushev).

**As per claim 15**, Bottomley ( combined with Driesen) does not teach creating maps for COM classes and interfaces; but Miloushev in a method to provide dispatch table similar to Driesen in a modeling environment, disclose alias table and COM/OLE interface with virtual table support ( Fig. 6, 19A; col. 56, para 7.9.5). If the Java environment by Bottomley support methods invocation in a multi-tiered modeling type development as in Miloushev, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide to Bottomley's Java dispatch table the dispatch/mapping table for dispatching of COM classes and interfaces as suggested by Miloushev because the use of COM and object remote invocation in a network paradigm using Java is a well-sought capability in network-based business and communication data retrieval at the time the invention was made.

**As per claim 21**, this claim corresponds to claim 15, hence is rejected using the rejection as set forth therein.

**As per claim 36**, refer to rejection of claim 15.

***Conclusion***

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Gagnon et al., "SableVM: A Research Framework for the Efficient Execution of Java Bytecode", Proceedings of the Java Virtual Machine Research ad Technology Symposium, 2-4, 2001, disclosing plurality of sparse array with empty slots.

U.S. Pat No. 6,393,491 to Bracha et al., disclosing vtable with primary index and increase of size if necessary.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

**Any response to this action should be mailed to**:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

**or faxed to:**

(703) 872-9306 ( for formal communications intended for entry)

**or:** (703) 746-8734 ( for informal or draft communications, please label "PROPOSED" or "DRAFT" – please consult Examiner before use)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor (Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

VAT
July 6, 20043

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100